

Zarooriyaat Platform

UNLOCKING E-COMMERCE FOR
EVERYONE



Problem Statement

01

Problem 01

Barrier to Entry for New
Buisnesses

02

Problem 02

Increased Financial Pressure

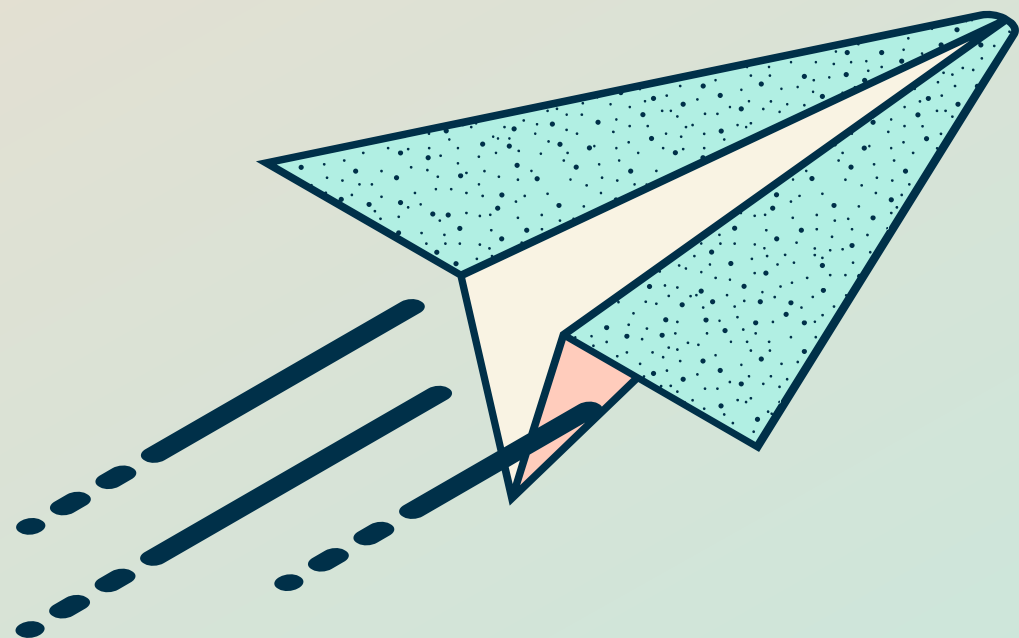
03

Problem 03

Reduced Flexibility for
Business Operations



Zarooriyaat Platform



01

Solution 01

No Upfront Costs

02

Solution 02

Performance-Based Fees

03

Solution 03

Scalability Support



Zarooriyaat Development

Transforming Vision into Value: From
Requirements Analysis, through Detailed
Design, to Robust Development





Interfaces

Registration

Full Name

Email

Phone Number


Password

Confirm Password

Register

Signup Form

Main Home Page

 Zarooriyaat

[Services](#) [About us](#)

Empower Your Business with Seamless Selling!

Join a platform designed to help you grow. Start selling smarter, faster, and without upfront fees!

Continue





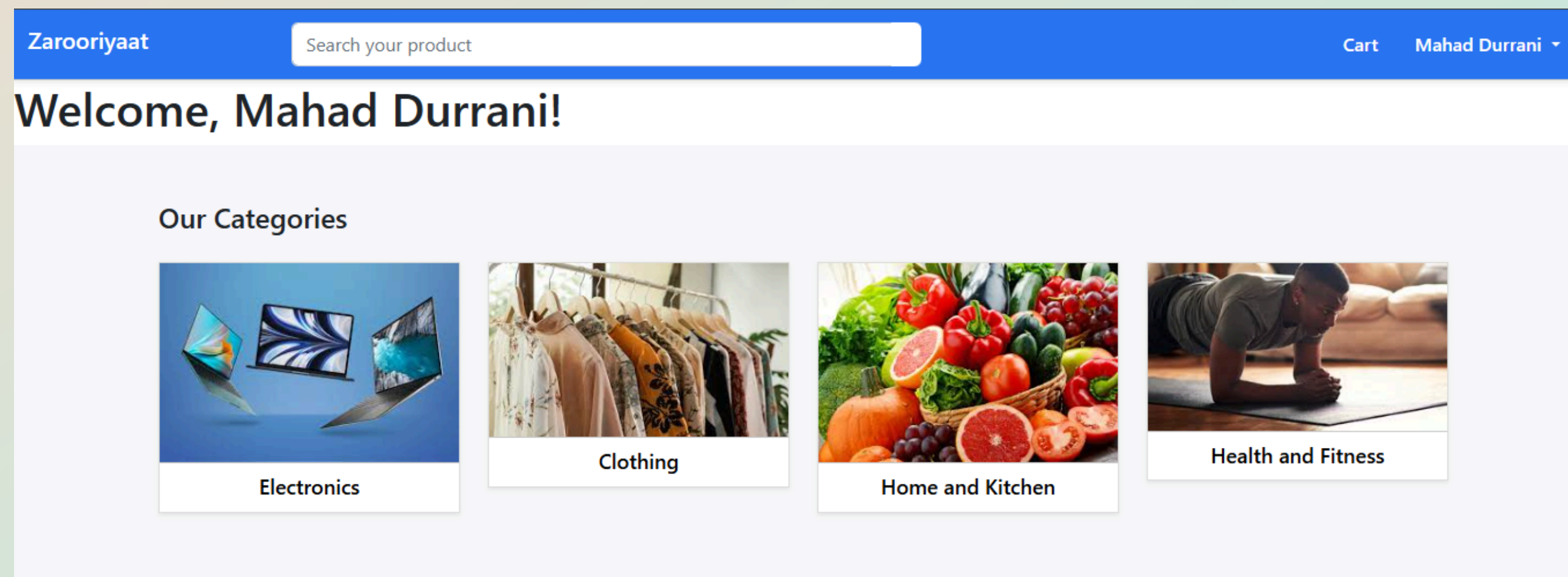
Interfaces

Login

☐ Remember me [Forgot password?](#)

[Not a member? Signup now!](#)

Login form



Categories Page



Interfaces

Control Panel

[Switch to Buying](#)

Seller Dashboard

Welcome null, Love to see you back.

[Add Products](#)[Remove Products](#)[Update Products](#)

Current Inventory:

Product ID	Product Name	Brand	Stock
------------	--------------	-------	-------

Zarooriyaat



Products in Category: Electronics

In Stock



HP

HP Laptop

500.0 ~~799.0~~

[Add To Cart](#)[View](#)

In Stock



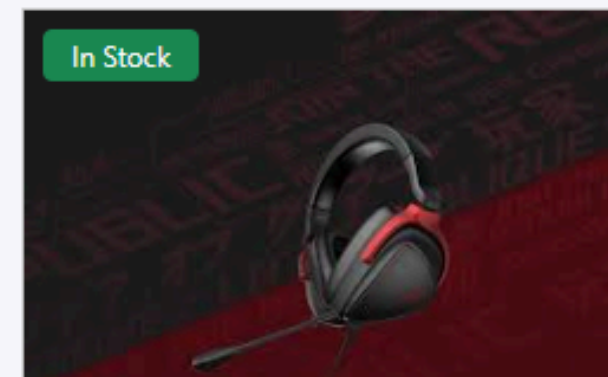
MI

Red MI Note 11

200.0 ~~300.0~~

[Add To Cart](#)[View](#)

In Stock



Asus

Head Phone

399.0 ~~499.0~~

[Add To Cart](#)[View](#)

Seller Dashboard



Interfaces

Checkout Summary

Red MI Note 11 - Rs. 200.00

Mangoes - Rs. 300.00

Total: Rs. 500.00

Enter your details and select a payment method:

Full Name

Mahad Durrani

Phone Number

03339131317

Address

H11, A.K Brohi Road, Islamabad, Pakistan

Select Payment Method:

Choose payment method...

Choose payment method...

Credit Card

PayPal

Checkout Page

Customer Dashboard

Customer Panel



☐ Dashboard

[Back to Dashboard](#)

Manage Products

Hello Mahad, manage your products here.

Review Product

Track Product

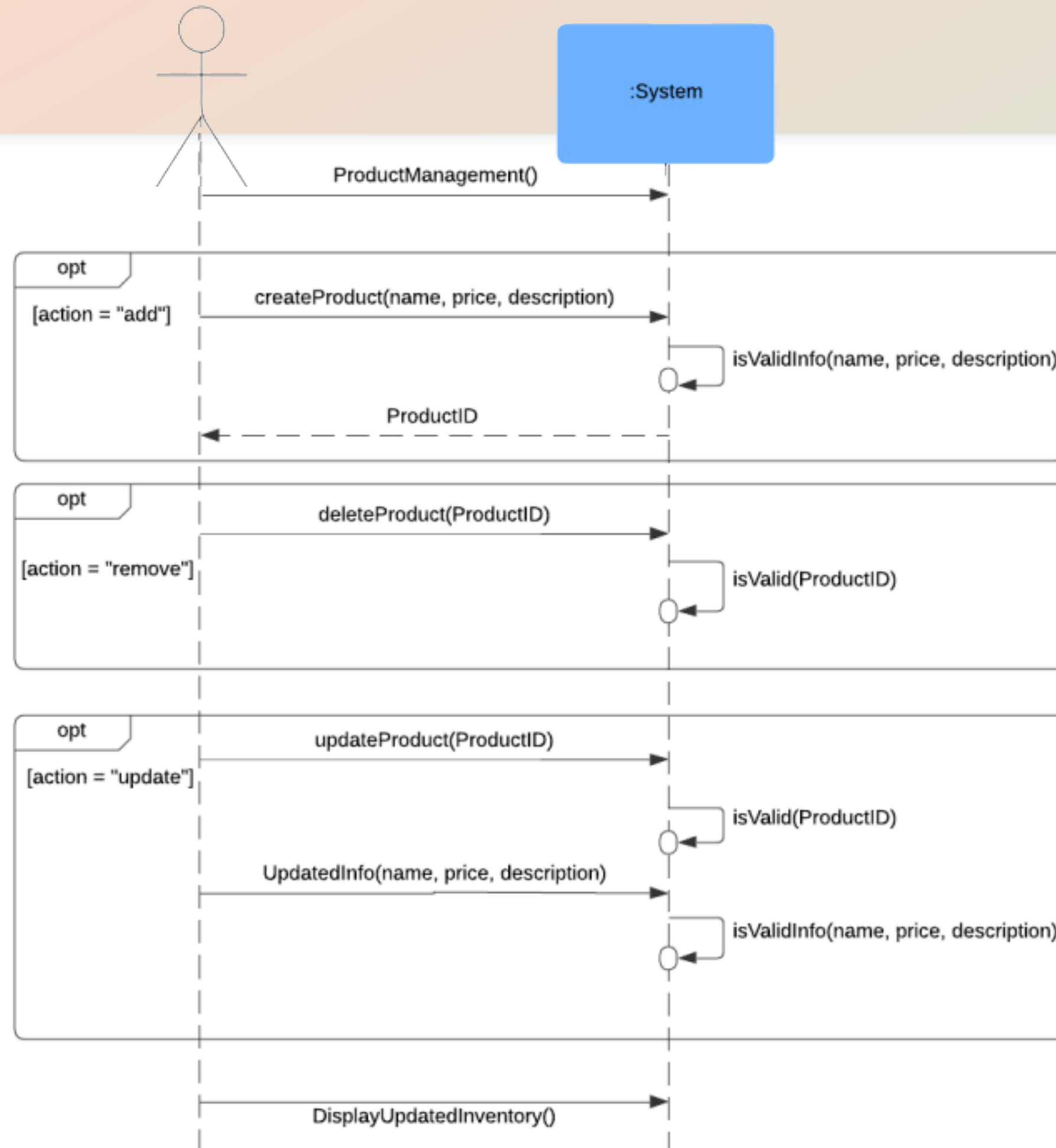
Return Product

Your Order History

Order ID	Product	Quantity	Total Price	Date
1	5	1	500.0 \$	2024-11-27T01:07:54.132519
2	2	1	500.0 \$	2024-11-27T01:07:54.279484

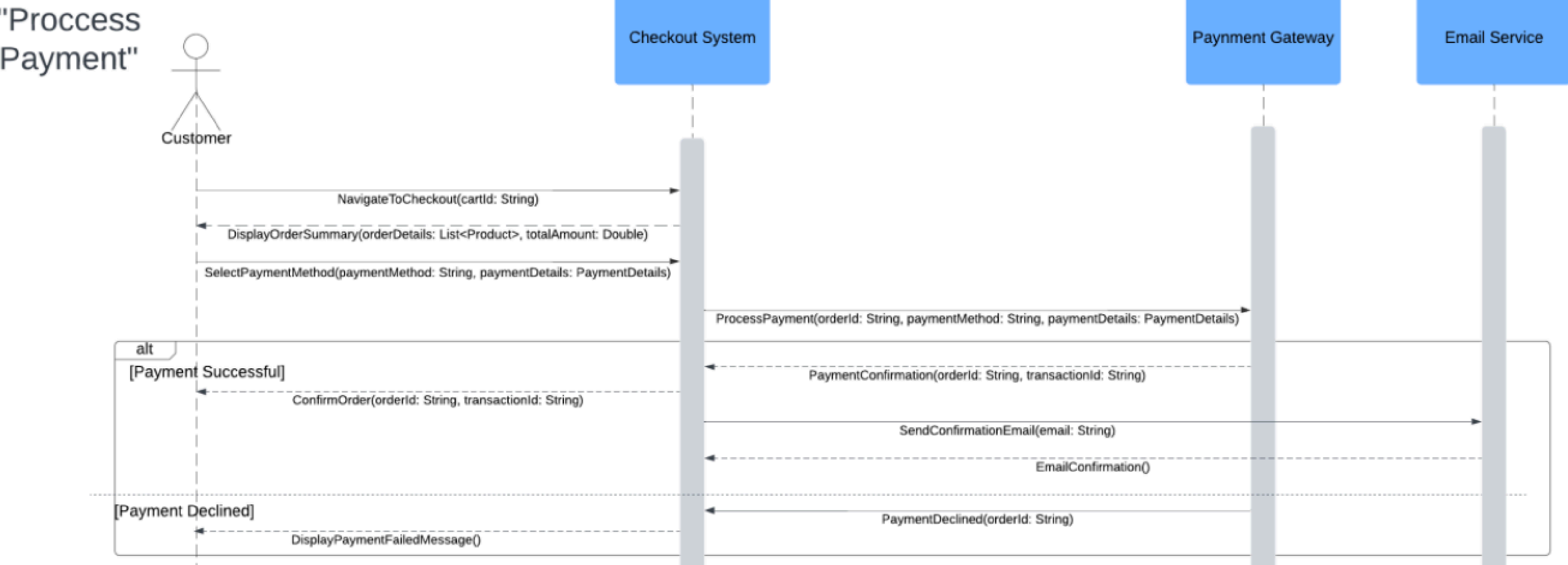
System Sequence Diagram

Manage Product Listing -
Usecase



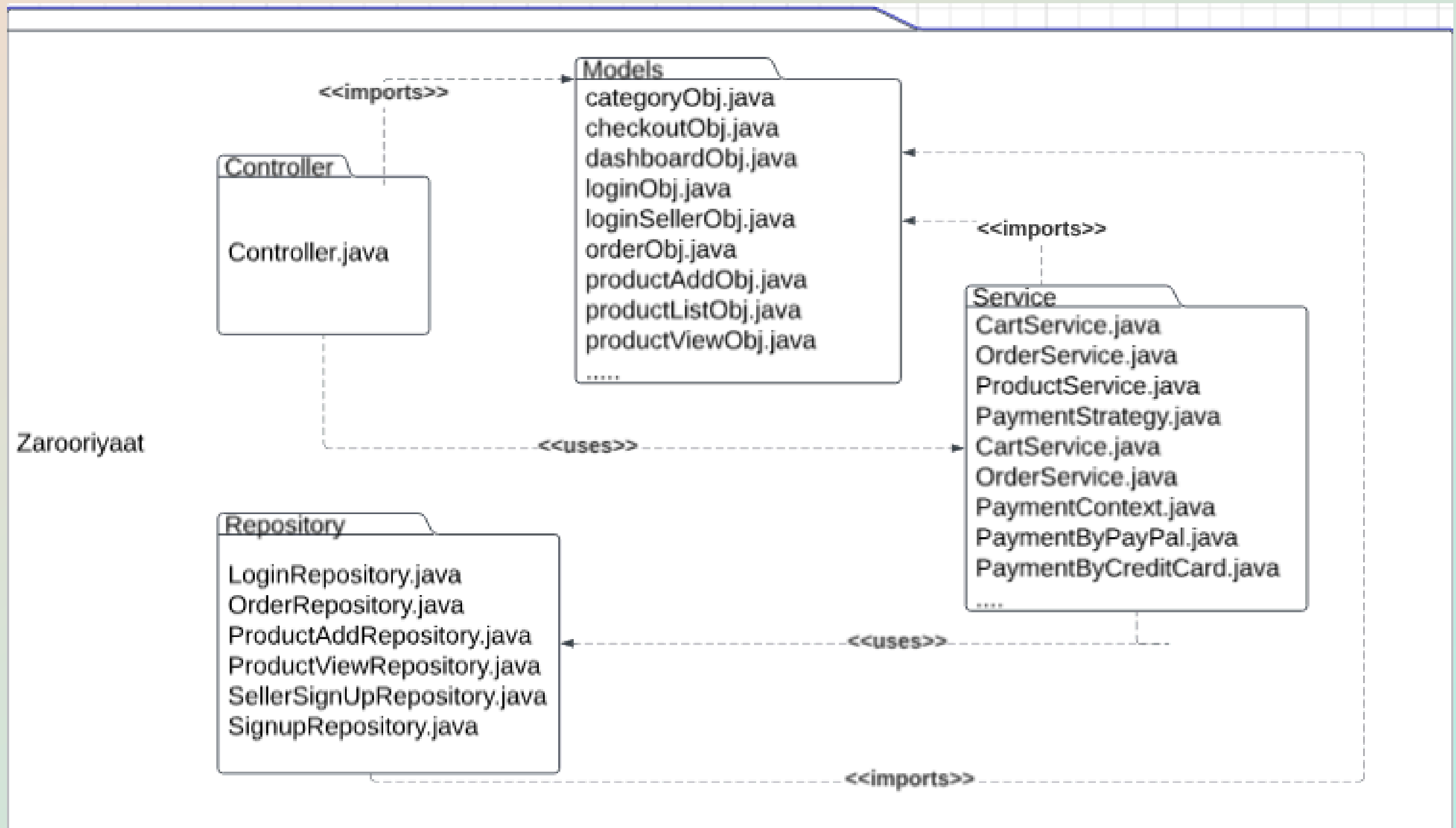
Action Sequence Diagram

Process Payment - Usecase



Package Diagram

An abstract view
of all the packages
involved



Design Patterns

GRASP Patterns

- Controller
- Creator
- High Cohesion
- Information Expert

GOF Patterns

- Strategy
- Repository

Controller

- Controller Pattern in GRASP, a Controller delegates tasks to appropriate domain and service objects

```
public class sdaController {  
  
    @GetMapping("/dashboard-seller/add-product")  
    public String showAddProductForm(Model model) {  
        model.addAttribute("product", new product());  
        return "addProduct";  
    }  
  
    @PostMapping("/dashboard-seller/add-product")  
    public String handleAddProduct(@ModelAttribute product product,  
        RedirectAttributes redirectAttributes) {  
        productAddRepository.save(product);  
        redirectAttributes.addFlashAttribute("message", "Product added successfully!");  
        return "redirect:/dashboard-seller";  
    }  
  
    @GetMapping("/removeProduct")  
    public String showRemoveProductPage() {  
        return "removeProduct";  
    }  
}
```


Creator


- Creator pattern is used for creating domain objects like Order or Product.

```
@Service
public class OrderService {

    @Autowired
    private OrderRepository orderRepository;

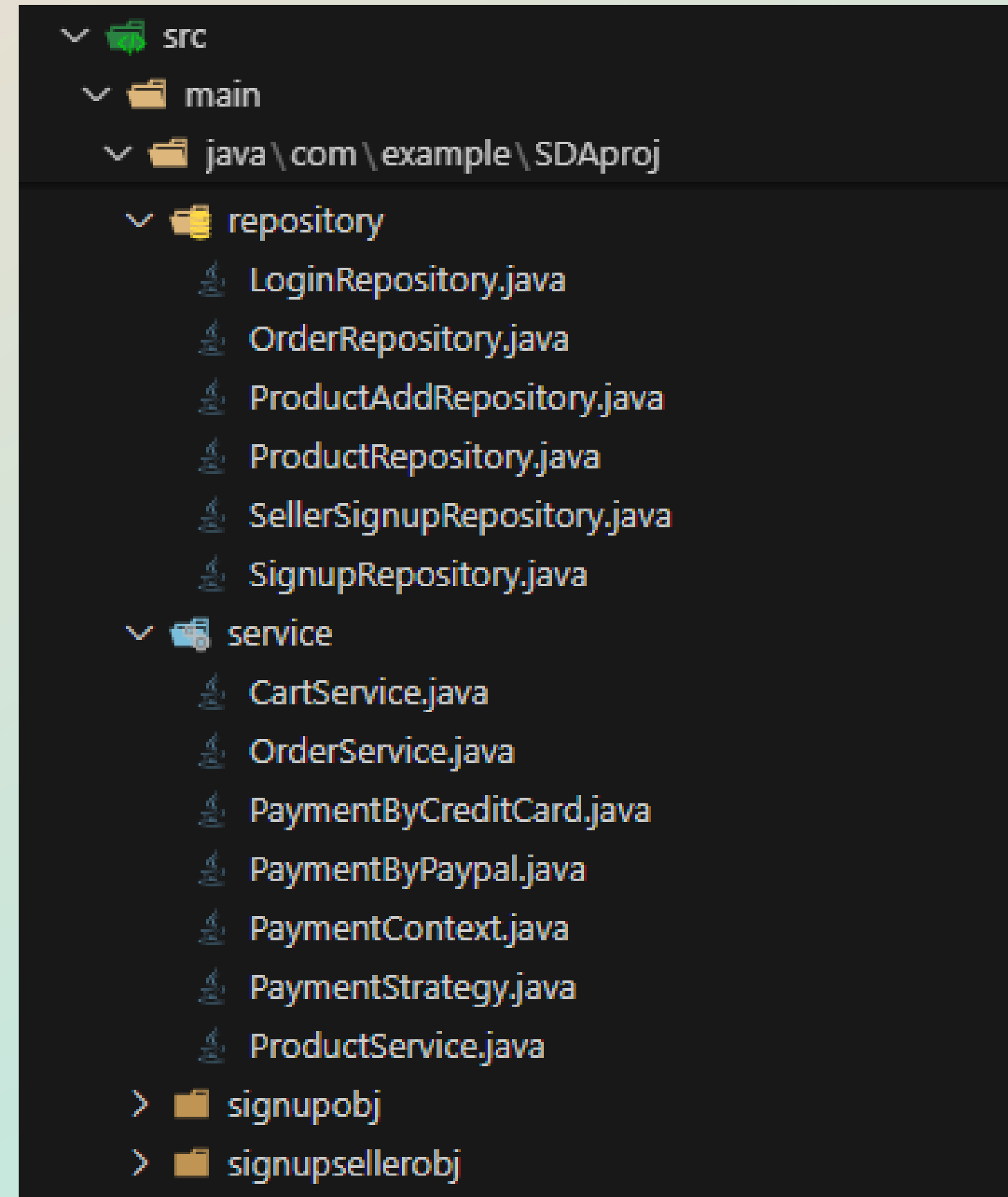
    public void saveOrder(List<productListObject> cartItems) {
        double totalPrice = cartItems.stream()
            .mapToDouble(productListObject::getSellingPrice)
            .sum();

        for (productListObject product : cartItems) {
            Order order = new Order();
            order.setProductId(product.getId());
            order.setQuantity(quantity:1);
            order.setTotalPrice(totalPrice);
            order.setOrderDate(LocalDateTime.now());
            order.setStatus(status:"placed");
            orderRepository.save(order);
        }
    }
}
```



High Cohesion

- By splitting responsibilities into distinct services and repositories, the system achieves high cohesion



Information Expert

- Domain classes encapsulate all data relevant to their domain.

```
package com.example.SDAproj.Orderobj;

import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table(name = "Orders")
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int orderId;

    @Column(nullable = false)
    private Long productId;

    @Column(nullable = false)
    private int quantity;
    @Column(nullable = false)
    private double totalPrice;

    @Column(nullable = false)
    private LocalDateTime orderDate;

    @Column(nullable = false)
    private String status;

    public int getOrderId() {
        return orderId;
    }
}
```

Strategy Pattern

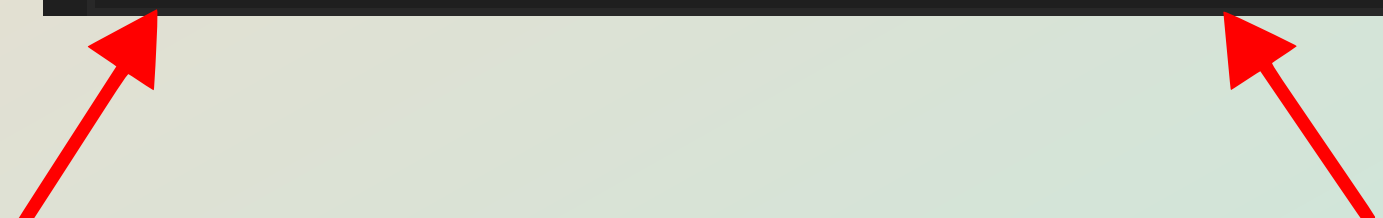
- The PaymentContext class is responsible for interacting with the selected strategy. Dynamically sets the desired strategy.

```
public class PaymentContext {  
    private PaymentStrategy paymentStrategy;  
  
    public void setPaymentStrategy(PaymentStrategy paymentStrategy)  
    {  
        this.paymentStrategy = paymentStrategy;  
    }  
  
    public String executePayment(double amount)  
    {  
        if (paymentStrategy == null) {  
            throw new IllegalStateException("Payment strategy is not set.");  
        }  
        return paymentStrategy.processPayment(amount);  
    }  
}
```

Strategy Pattern

```
package com.example.SDAproj.service;

public interface PaymentStrategy
{
    String processPayment(double amount);
}
```



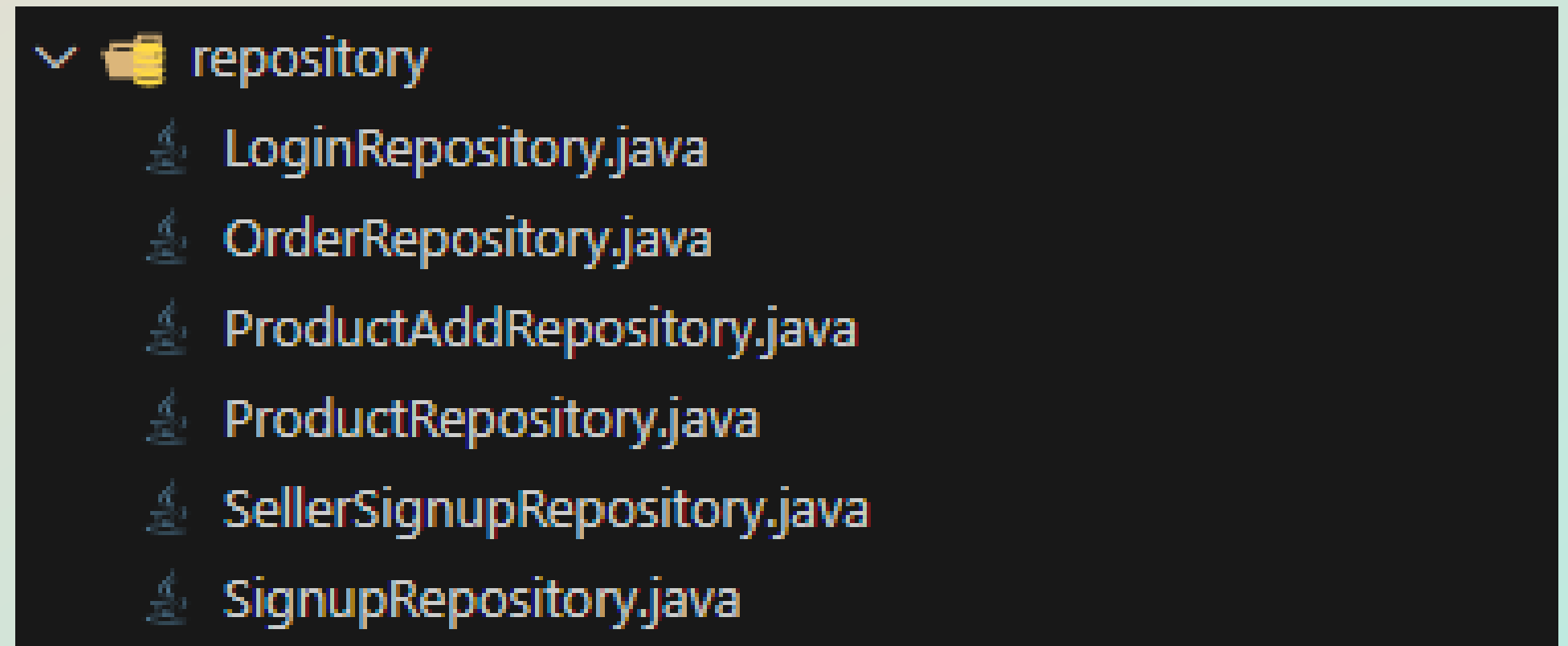
Two red arrows point from the bottom of the two concrete class code blocks to the PaymentStrategy interface block, indicating that both classes implement the interface.

```
public class PaymentByCreditCard implements PaymentStrategy
{
    @Override
    public String processPayment(double amount) {
        return "Payment of " +
            amount + " processed via Credit Card.";
    }
}
```

```
public class PaymentByPaypal implements PaymentStrategy
{
    @Override
    public String processPayment(double amount)
    {
        return "Payment of "
            + amount + " processed via PayPal.";
    }
}
```

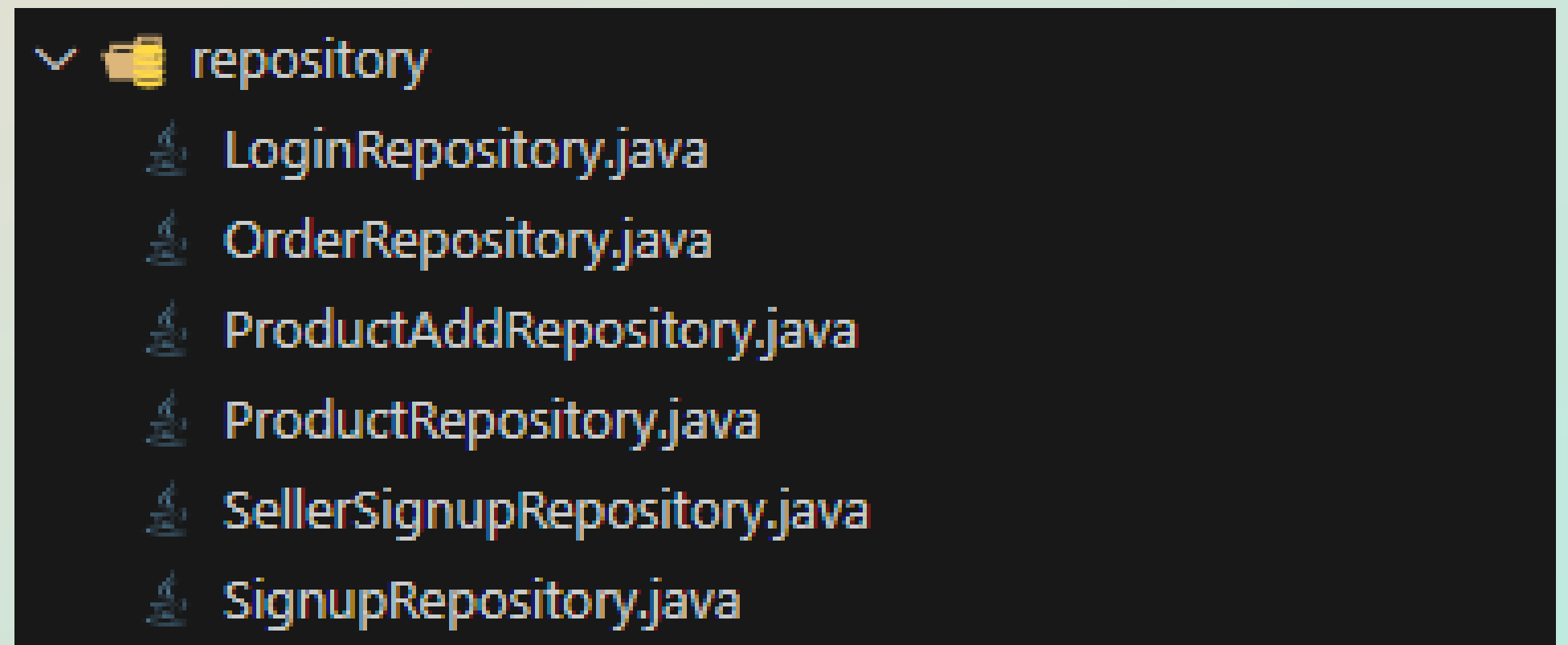

Repository Pattern

- The repository layer in the project abstracts the data access logic by encapsulating queries and interactions with the database.



Repository Pattern

- The repository layer in the project abstracts the data access logic by encapsulating queries and interactions with the database.



Meet the Team



Mahad Rehman

Scrum Master



Hashim Awan

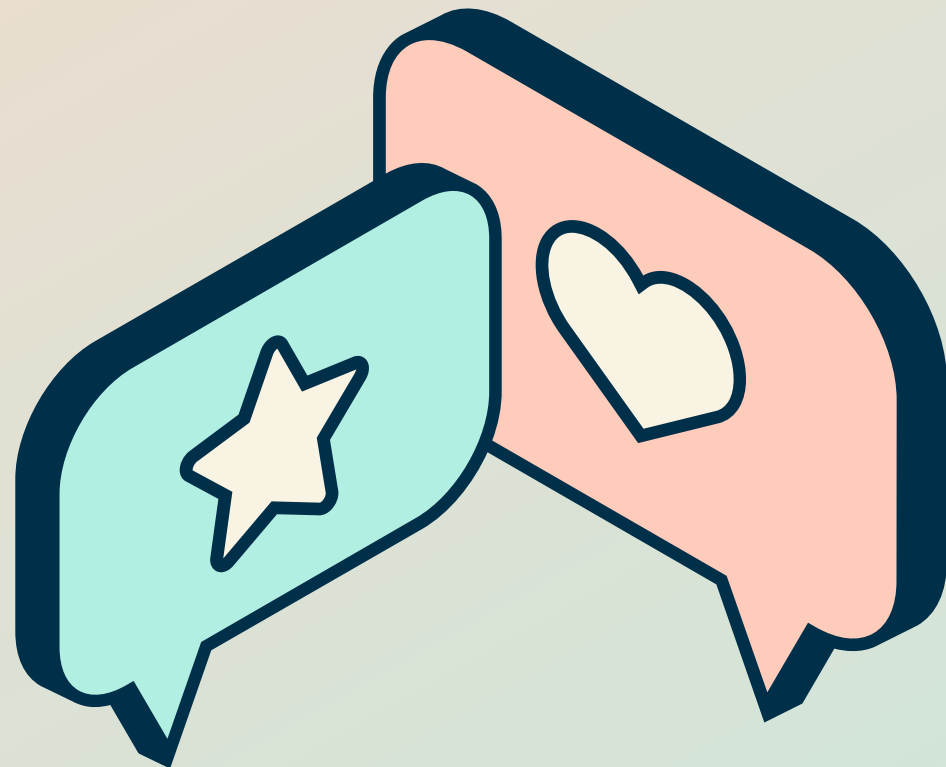
Senior Developer



Masab Hammad

Senior Developer

THANK YOU!



Email Address

zarooriyaat@gmail.com



Phone Number

+92 333 9131317



Mailing Address

AK. Brohi Road, H11/1